

# Tree Memory Networks for Modelling Long-term Temporal Dependencies

**Tharindu Fernando**

*Image and Video Research Laboratory, SAIVT  
Queensland University of Technology,  
Australia.*

T.WARNAKULASUIRYA@QUT.EDU.AU

**Simon Denman**

*Image and Video Research Laboratory, SAIVT  
Queensland University of Technology,  
Australia.*

S.DENMAN@QUT.EDU.AU

**Aaron McFadyen**

*Robotics and Autonomous Systems,  
Queensland University of Technology,  
Australia.*

AARON.MCFADYEN@QUT.EDU.AU

**Sridha Sridharan**

*Image and Video Research Laboratory, SAIVT  
Queensland University of Technology,  
Australia.*

S.SRIDHARAN@QUT.EDU.AU

**Clinton Fookes**

*Image and Video Research Laboratory, SAIVT  
Queensland University of Technology,  
Australia.*

C.FOOKES@QUT.EDU.AU

**Editor:**

## Abstract

In the domain of sequence modelling, Recurrent Neural Networks (RNN) have been capable of achieving impressive results in a variety of application areas including visual question answering, part-of-speech tagging and machine translation. However this success in modelling short term dependencies has not successfully transitioned to application areas such as trajectory prediction, which require capturing both short term and long term relationships. In this paper, we propose a Tree Memory Network (TMN) for modelling long term and short term relationships in sequence-to-sequence mapping problems. The proposed network architecture is composed of an input module, controller and a memory module. In contrast to related literature, which models the memory as a sequence of historical states, we model the memory as a recursive tree structure. This structure more effectively captures temporal dependencies across both short term and long term sequences using its hierarchical structure. We demonstrate the effectiveness and flexibility of the proposed TMN in two practical problems, aircraft trajectory modelling and pedestrian trajectory modelling in a surveillance setting, and in both cases we outperform the current state-of-the-art. Furthermore, we perform an in depth analysis on the evolution of the memory module content over time and provide visual evidence on how the proposed TMN is able to map both long term and short term relationships efficiently via a hierarchi-

cal structure.

**Keywords:** Memory Networks, Trajectory Prediction, Recurrent Neural Networks

## 1. Introduction

Sequence-to-sequence modelling is a vital element in machine learning and knowledge representation with multiple application areas including machine translation (ShafieiBavani et al., 2016), trajectory prediction (Judah et al., 2014), and part-of-speech tagging (Lafferty et al., 2001). This problem can be represented as predicting an output sequence,  $Y = [y_1, \dots, y_T]$ , given an input sequence  $X = [x_1, \dots, x_T]$ . Predicting a future element  $y_t$  of the sequence at time instance  $t$  utilising the current input to the model  $x_t$  and the content of the memory from the previous time step  $M_{t-1}$  can be represented as,

$$y_t = f(x_t, M_{t-1}). \quad (1)$$

Modelling long term relationships within sequences can be considered one of the most challenging problems within the machine learning community (Mikolov et al., 2014). Although many memory architectures proposed for sequence-to-sequence modelling are capable of mapping short term relationships, they are less successful when handling long term dependencies (Xu et al. (2016)).

Long term relationships within the data are extremely useful and can significantly influence the accuracy of the predictions considering the repetitive nature of many processes. For instance consider an air traffic modelling problem. Even though short term dependencies such as current weather and neighbouring traffic are the most influential factors, the repetitive nature of the aircraft trajectories and the flight schedules suggests that one can easily deduce a coherence among trajectories between days and seasons. A certain runway may be in use, so similar trajectories should be observed over the short term, but a sudden change of runway (due to weather for example) means “new” trajectories will be seen. Having a long term memory means that these are not actually “new” trajectories, but can instead be recalled from an earlier, similar weather event or operational configuration.

A similar logic can be applied when modelling pedestrian behaviour in a surveillance scenario, such as in the example shown in Fig. 1. While the current location of neighbouring pedestrians are more influential, one cannot discard the influence of historical behaviour under similar contexts and events. Pedestrians may be wandering in a free area and then as new trains arrive a new “flow” of pedestrian movement appears in response to the congestion. But as we possess historical data under similar contexts one should be capable of accurately anticipating such pedestrian motion.

In this paper we are interested in efficiently aggregating such long term dependencies among input data. In sample scenario from the Grand Central dataset (Yi et al. (2015)) presented in Fig. 1, a high pedestrian flow in the highlighted area is observed at the 07:08 time stamp with the arrival of a train before the flow decreases at the 10:32 time stamp. A similar flow to Fig. 1 (a) is observed at the 21:29 time stamp, in Fig. 1 (c) confirming our hypothesis that future pedestrian flow can be anticipated with the aid of historical data.

As such, we propose an augmented memory architecture which can be generalised to any sequence to sequence modelling problem. The contributions of this work can be summarised as follows:

1. A new recursive memory network architecture capable of modelling long term temporal dependencies, using an efficient tree structure.

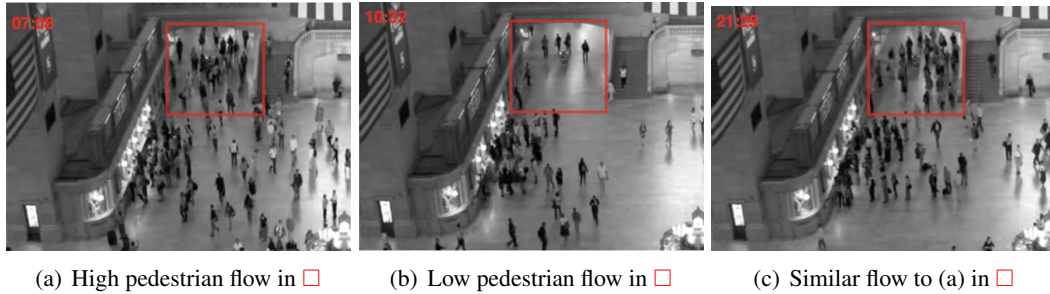


Figure 1: Pedestrian flow at different times in the Grand central dataset Yi et al. (2015). A high pedestrian flow in the highlighted area is observed at the 07:08 time stamp with the arrival of a train before the flow decreases at the 10:32 time stamp. A similar flow to subfigure (a) is observed at the 21:29 time stamp confirming our hypothesis that future pedestrian flow can be anticipated with the aid of historical data.

2. Application of the proposed memory architecture to two practical problems: aircraft trajectory modelling and pedestrian trajectory modelling in a surveillance setting, where in both cases we are able to achieve state-of-the-art results.
3. An in depth analysis on the evolution of the memory module content where we study the changes in hidden state representations over time and elaborate on some interpretable patterns.

The two applications we demonstrate the proposed approach on, aircraft trajectory prediction and pedestrian trajectory prediction, are both related sequence-to-sequence modelling tasks, however they have quite distinct characteristics that illustrate the adaptability of the proposed approach. Aircraft trajectories are primarily a function of the flight schedule, which is typically fixed over a week, but still varies according to changes such as weather, off schedule arrivals/departures, etc. Pedestrian trajectories however are less dependent on a schedule and are more influenced by the behaviour of other nearby pedestrians.

We would like to emphasise the fact that even though we are demonstrating our approach on two different application scenarios from the trajectory prediction domain, the varied nature of these problems demonstrates how the proposed model can be directly applied to any sequence-to-sequence prediction problem where modelling long term relationships is necessary. Possible application areas include diver behaviour modelling for autonomous driving (Chen et al. (2015); Liu et al. (2017)), text and video synthesis (Reed et al. (2016)), and context aware machine translation (Banchs (2014)).

## 2. Related works

Related work within the scope of this paper can be categorised into memory architectures (in Section 2.1), aircraft trajectory prediction approaches (in Section 2.2) and pedestrian trajectory prediction approaches (in Section 2.3).

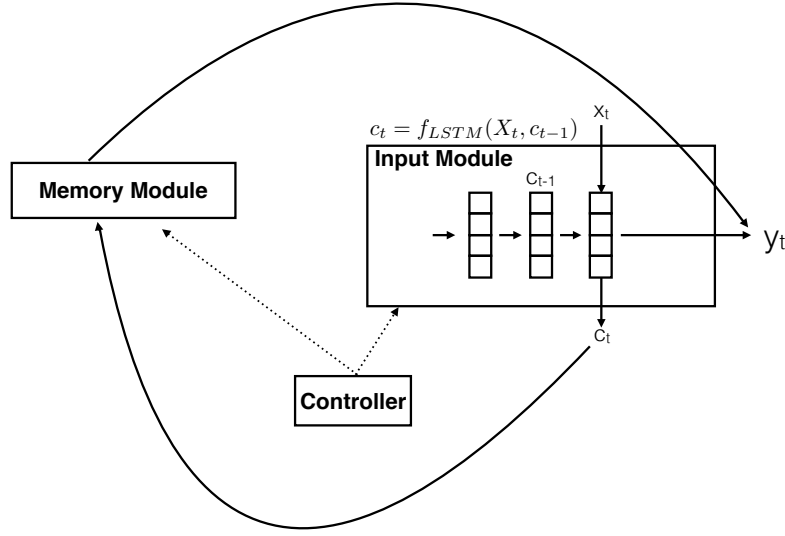
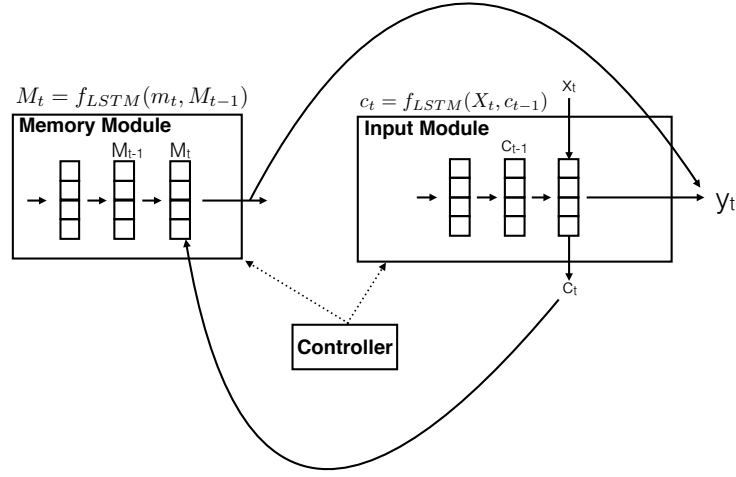


Figure 2: Neural Network architecture with an external memory. The memory is used to store important historical facts which can be utilised for future predictions. The controller is responsible for issuing read and write commands in order to take out and write back to the memory. An input module is used to encode and generate a vector representation from the input

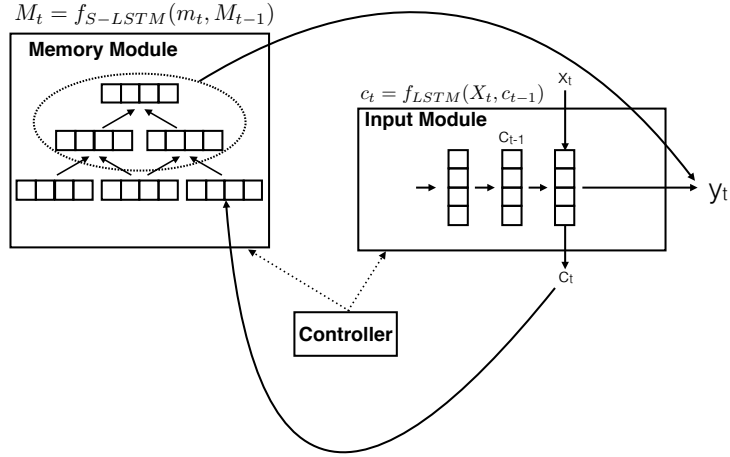
## 2.1 Memory architectures

Deep learning models such as Recurrent Neural Networks (RNN) have been applied extensively for many sequence-to-sequence modelling problems and have been capable of producing state-of-the-art results. A number of approaches (Weston et al. (2014); Kumar et al. (2015); Malinowski and Fritz (2014); Joulin and Mikolov (2015)) have also utilised what are termed “memory modules”, to aid prediction. The memory stores important facts from historical inputs and then generates the future predictions based on the stored knowledge. A sample architecture with an input module, controller and an external memory is shown in Fig. 2. Firstly the input module generates a vector representation  $c_t$  for the input  $x_t$  at time instance  $t$ . The controller then triggers a memory read operation. The memory module, with an attention process, searches the history and outputs relevant facts. The final output is generated by merging  $c_t$  with the memory output. Finally, the controller triggers a memory update operation where the memory  $M_{t-1}$  is updated with  $c_t$ .

Weston et al. (2014) have utilised a memory module to improve the accuracy on natural language processing problems. Their proposed memory architecture is not fully extendible considering the usage of an offline feature engineering process using a bag-of-words approach. In similar works, such as Joulin and Mikolov (2015) and Kaiser and Sutskever (2015) for image caption generation, and Malinowski and Fritz (2014) and Chen and Zitnick (2014) for visual question answering, the authors have extensively applied the notion of external memory. The memory architecture, “episodic memory”, proposed in Kumar et al. (2015) has been shown to be capable of outperforming the other external memory architectures proposed above in terms of accuracy .



(a) Episodic memory model proposed in Kumar et al. (2015) with LSTM memory cells



(b) Proposed TMN model with S-LSTM memory cells.

Figure 3: Comparison of the memory model proposed in Kumar et al. (2015) (a) with the proposed memory model (b). In both approaches at time instance  $t$ , the input module generates representation  $c_t$  for the input  $x_t$ . Then the controller triggers a memory read operation. The memory module, with the attention process, output the relevant facts. The final output is given by merging  $c_t$  with the memory output. Finally the memory update operation updates memory  $M_{t-1}$  with  $c_t$ .

Fig. 3 (a) depicts the episodic memory model proposed in Kumar et al. (2015). The authors model the “episodic memory” as a hierarchical recurrent sequence model utilising the sequential nature of the memory. The authors propose a generalised neural sequential module with recurrent LSTM memory cells for sequence encoding, memory mechanism and response generation. The above work is further extended with a shared memory architecture in Neelakantan et al. (2015).

Even with the exemplary results for short term dependency modelling problems, none of the above stated architectures are capable of handling sequences with long term relationships.

In approaches such as (Kumar et al., 2015; Neelakantan et al., 2015; Munkhdalai and Yu, 2016a) the memory is composed of a single layer of memory units. The memory update mechanism in (Kumar et al., 2015; Neelakantan et al., 2015) can be written as,

$$\mathbf{M}_t = f_{LSTM}(\mathbf{m}_t, \mathbf{M}_{t-1}), \quad (2)$$

where  $\mathbf{m}_t$  is a score value that quantifies the relevance of the content of the memory module ( $\mathbf{M}_{t-1}$ ) at time  $t-1$  to the current context,  $\mathbf{c}_t$  (see 3 (a)); where as in Munkhdalai and Yu (2016a) the authors completely update the content of the memory locations based on  $\mathbf{m}_t$ .

The main drawback of using approaches such as (Kumar et al. (2015); Neelakantan et al. (2015); Munkhdalai and Yu (2016a)) with long term dependency modelling and big data sets is that, in most cases, the attention mechanism will generate small scores for all the examples as they are all dissimilar to the input. Therefore with those approaches we are required to maintain an extremely large memory sequence as similar inputs only occur after long time intervals. Furthermore, in recurrent models such as in LSTMs, when the sequence becomes too long the output becomes more biased towards recent historical states (Chen et al. (2016)) rather than considering the entire set of historical states equally.

Hypothetically if the memory module has enough non-linearity and if we have a sufficiently large database, any output should be able to be produced by the components that reside in the memory. Furthermore the model should be capable of learning and modelling the short term contextual effects as well as the long term contextual effects from what is input to its memory. This can be seen as a dictionary learning process where memory is the dictionary that we are learning and the memory module should learn to produce an accurate prediction for different combinations of inputs and historical trajectories.

Recently the recursive LSTM (S-LSTM) (Zhu et al. (2015)) structure was proposed where the authors extend the sequential LSTM to tree structures, in which a memory cell can reflect the historical memories of multiple child cells or multiple descendant cells in a recursive process. Chen et al. (2016) have performed an in depth analysis of the strengths and weaknesses of sequential and recursive structures for a neural language modelling task; and found syntactical relationships such as structure and logic in the input data are best captured via a recursive LSTM structure (i.e. S-LSTM), where as when encoding the semantics of sentences, sequential LSTM models provide state-of-the-art results.

The proposed model is illustrated in Fig 3 (b). Our approach eschews the sequence representation in favour of a tree-based approach, which is motivated by the positive characteristics that the S-LSTM architecture exhibits such as feature compression power and the preservation of semantic relationships among data. A detailed analysis of the architecture in comparison to state-of-the-art methods is provided in Section 3.3.

## 2.2 Aircraft trajectory prediction approaches

Approaches such as (Prandini et al., 2000; Paielli and Erzberger, 1997) utilise probability models on aircraft dynamics in order to generate predictions on their future motion. They purely rely on the assumptions made on the dynamics of the aircraft, without using any historical information, which contributes to its main drawback.

In Choi and Hebert (2006), de Leege et al. (2013), Winder (2004) researchers treated aircraft trajectory prediction as a machine learning problem, in which they train the model using historical trajectory data together with weather observations. Most recently Ayhan and Samet (2016) proposed an approach that considered trajectories as a set of 4 dimensional data cubes, together with weather parameters. Initially they performed time series clustering on data for segmentation and then learnt a HMM on top of each cluster. Still due to the uncertainty with weather observations, these trajectory prediction approaches become inefficient.

Several efforts have been made to improve the trajectory prediction by better wind estimation (Mondoloni and Liang (2003); Cole et al. (2000); Rekkas et al. (1991); Delahaye (1992); Hollister et al. (1989)), yet these approaches have failed to achieve significant improvement in the task of trajectory prediction. Furthermore we would like to emphasise the fact that all of the above stated approaches consider aircrafts individually without considering the air traffic within the neighbourhood, completely discarding important factors such as the volume and proximity of nearby air traffic. Even though weather is a vital factor for future predictions it is implicit in neighbouring traffic behaviour. Therefore, inferring a notion of weather through neighbouring traffic is computationally inexpensive compared to tedious interpolations that ground based weather observations require (Ayhan and Samet, 2016).

### 2.3 Pedestrian trajectory prediction approaches

When considering literature for human behaviour prediction the social force model (Helbing and Molnár (1995); Koppula and Saxena (2013); Pellegrini et al. (2010); Yamaguchi et al. (2011); Xu et al. (2012)) and its variants can be considered as well-established. Such approaches generate attractive and repulsive forces between pedestrians defining the optimal path under different contexts with respect to the neighbourhood. As a variant, a mixture model is presented in Zhou et al. (2015) but this approach ignores the interactions among pedestrians. Wang et al. (2008) proposed a “topic model” which was extended to incorporate spatio-temporal dependencies in Hospedales et al. (2009) and Emonet et al. (2011). All of the above stated approaches use hand-engineered features as the input to the prediction module, which can be considered their main drawback as they fail to account for the semantics of the scene. Depending on the domain knowledge of the feature engineer, hand-crafted features may only capture abstract level semantics of the environment.

Alahi et al. (2016) removed the need for hand-crafted features via an unsupervised feature learning approach. The authors encode the trajectory of each pedestrian in the scene at that particular time frame using LSTMs. The hidden states of the neighbouring pedestrians in the immediately preceding time step are used for generating their positions in the current time step. As pointed out in Fernando et al. (2017b) this approach is only sufficient for generating reactive behaviours such as collision avoidance but fails to generate smooth trajectories for long term trajectory planning. Fernando et al. (2017b) have extended the above idea to incorporate the entire trajectory of the pedestrian of interest as well as the neighbouring pedestrians. To the best of our knowledge none of the literature addressing human behaviour prediction has considered the long-term relationships among human behavioural patterns. Motivated by this limitation, we intend to explore the utility of temporal data for trajectory prediction via a tree memory network.

### 3. Tree Memory Network (TMN) Model

In this work, we are motivated by the exemplary results that were achieved from a tree structure for the task of discriminative dictionary learning from trajectories in Fernando et al. (2017a). In contrast to mapping all the historic data with a shallow layer of recurrent memory cells, we hierarchically map the memory with a bottom up tree structure where all the historical states are represented in the bottom layer of the tree and we go up the hierarchy concatenating the most significant features in order to generate the output at a particular time step.

Furthermore, rather than stacking individual recurrent layers (such as in Serban et al. (2016); Li et al. (2015)), we utilise a Tree-LSTM structure as it focusses only on the historical information from its two neighbours. Therefore it can be seen as passing out significant features from two temporally adjacent neighbours to the upper layer.

#### 3.1 Input Module

Let  $X_i = [\mathbf{x}_i^1, \mathbf{x}_i^2, \dots, \mathbf{x}_i^T]$  be the  $i^{th}$  input sequence where  $T$  represents the number of time steps. The input module computes a vector representation  $\mathbf{c}_t$  for the input sequence via a LSTM layer,

$$\mathbf{c}_t = f_{LSTM}(\mathbf{x}_t, \mathbf{c}_{t-1}), \quad (3)$$

where  $\mathbf{c}_t \in \mathbb{R}^k$ , and  $k$  is the embedding dimension of the LSTM.

#### 3.2 Memory Module

Consider  $N \in \mathbb{R}^{p \times k}$  as the sequence of historical LSTM embeddings, with length  $p$  and embedding dimension  $k$ , that we would like to model as the memory. It can be seen as a queue structure with length  $p$  and each data element within the queue has a dimension of  $k$ . We adapt the S-LSTM model to represent the memory module of the proposed framework. It extends the general sequential structure in LSTMs to a bottom up tree structure composed of a compressed representation of children nodes to a parent node. This provides us with a principled way of considering long-distance interactions between memory inputs, and avoids the current drawbacks of LSTM models when handling lengthy sequences (Chen et al. (2016)). In order to preserve the simplicity we only represent the recursive-LSTM structure as a binary tree, where each parent node has a maximum of two child nodes; however the extension of this model to any tree structure is straightforward.

##### 3.2.1 MEMORY READ

When computing an output at time instance  $t$  we extract out the tree configuration at time instance  $t-1$ . Let  $M_{t-1} \in \mathbb{R}^{k \times (2^{l-1})}$  be the memory matrix resultant from concatenating nodes from the tree from the top to  $l = [1, \dots]$  depth. This allow us to capture different levels of abstraction that exists in our memory network. Let  $f^{score}$  be an attention scoring function which can be implemented as a multi-layer perceptron (Munkhdalai and Yu (2016b)),

$$\mathbf{m}_t = f^{score}(\mathbf{M}_{t-1}, \mathbf{c}_t), \quad (4)$$

$$\alpha = softmax(\mathbf{m}_t). \quad (5)$$



Eq. 4 and Eq. 5 provide an attention mechanism that finds the most relevant memory items given the current input,

$$\mathbf{z}_t = \mathbf{M}_{t-1} \alpha^T. \quad (6)$$

Then the final output can be represented as,

$$y_t = \text{ReLU}(W_{out} \mathbf{z}_t + (1 - W_{out}) \mathbf{c}_t), \quad (7)$$

where  $W_{out}$  are the output weights.

### 3.2.2 MEMORY UPDATE

In the proposed memory architecture each memory cell contains one input gate  $i_t$ , one output gate  $o_t$  and two forget gates  $f_t^L$  and  $f_t^R$ . At time instance  $t$  each node in the memory network can be updated in the following manner,

$$i_t = \sigma(W_{hi}^L h_{t-1}^L + W_{hi}^R h_{t-1}^R + W_{ci}^L c_{t-1}^L + W_{ci}^R c_{t-1}^R), \quad (8)$$

$$f_t^L = \sigma(W_{hf_l}^L h_{t-1}^L + W_{hf_l}^R h_{t-1}^R + W_{cf_l}^L c_{t-1}^L + W_{cf_l}^R c_{t-1}^R), \quad (9)$$

$$f_t^R = \sigma(W_{hf_r}^L h_{t-1}^L + W_{hf_r}^R h_{t-1}^R + W_{cf_r}^L c_{t-1}^L + W_{cf_r}^R c_{t-1}^R), \quad (10)$$

$$\beta = W_{hc}^L h_{t-1}^L + W_{hc}^R h_{t-1}^R, \quad (11)$$

$$c_t^P = f_t^L \otimes c_{t-1}^L + f_t^R \otimes c_{t-1}^R + i_t \otimes \tanh(\beta), \quad (12)$$

$$o_t = \sigma(W_{ho}^L h_{t-1}^L + W_{ho}^R h_{t-1}^R + W_{co}^P c_t^P), \quad (13)$$

$$h_t^P = o_t \otimes \tanh(c_t^P), \quad (14)$$

where  $h_{t-1}^L$ ,  $h_{t-1}^R$ ,  $c_{t-1}^L$  and  $c_{t-1}^R$  are the hidden vector representations and cell states of the left and right children respectively. The relevant weight vectors  $W$  are represented with appropriate super and subscripts where the superscript represents the relevant child node, and the subscript represents the relevant gate and the vector which the weight is attached to. The process is also illustrated in Fig 4.

### 3.3 Relation to current state of the art

The major difference between the proposed approach and current state of the art methods such as Kumar et al. (2015), Neelakantan et al. (2015) and Munkhdalai and Yu (2016a) is the representation of the memory. In those approaches the memory is composed of a single layer of memory units. The memory update mechanism in Kumar et al. (2015) and Neelakantan et al. (2015) can be written as,

$$\mathbf{M}_t = \text{LSTM}(\mathbf{m}_t, \mathbf{M}_{t-1}), \quad (15)$$

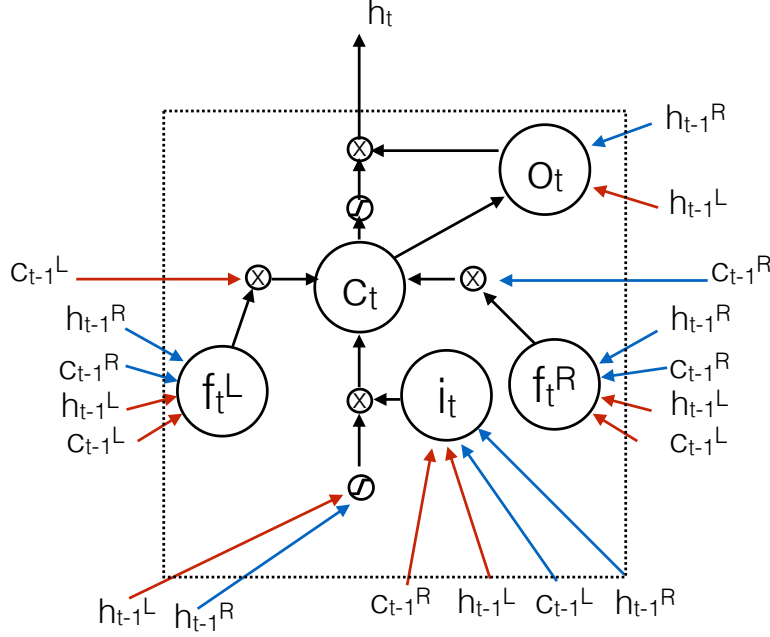


Figure 4: Tree memory cell architecture.  $f_t^L, f_t^R, o_t, i_t$  represents the left forget gate, right forget gate, output gate and input gate respectively.  $\otimes$  represents multiplication

where  $m_t$  can be obtained using Eq. 4, where as in Munkhdalai and Yu (2016a) the authors completely update the content of the memory locations with respect to the output of Eq. 6. This process can be written as,

$$\mathbf{M}_t = \mathbf{M}_{t-1}(1 - (\mathbf{z}_t \otimes \mathbf{e}_k)^T) + (h_t \otimes \mathbf{e}_p)(\mathbf{z}_t \otimes \mathbf{e}_k)^T, \quad (16)$$

where 1 is a matrix of ones and  $e_k$  and  $e_p$  are vectors of ones.  $\otimes$  denotes the outer product which duplicates its left vector  $p$  or  $k$  times to form a matrix. In contrast we model our memory in a binary tree structure where it is updated in a bottom up fashion utilising Eq. 8 to 14.

The main drawback of using approaches such as (Kumar et al. (2015); Neelakantan et al. (2015); Munkhdalai and Yu (2016a)) with long term dependency modelling and big data sets is that, in most cases, the attention mechanism will generate small score values for all the examples as they are dissimilar to the input. Therefore with those approaches we are required to maintain an extremely large memory sequence as similar inputs only occur after long time intervals. Furthermore in recurrent models such as in LSTMs, when the sequence becomes too long the output becomes more biased towards recent historic states (Chen et al. (2016)) rather than considering the entire set of historic states equally. In contrast, we represent memory with a tree structure where we learn the logical coherence among neighbouring memory cells with different levels of abstraction.

## 4. Experimental results

We present the experimental results on two trajectory datasets: an aircraft trajectory database from the south east Queensland (SEQ) region in Australia; and a widely utilised pedestrian trajectory database. These datasets are specifically chosen to demonstrate the capability of the proposed model to handle varying dimensionalities and temporal relationships and present its real world applicability.

### 4.1 Experiment 1: Terminal Area Air Traffic Prediction

We obtain air traffic data from the south east Queensland (SEQ) region in Australia from 30-11-2014 to 30-11-2015. We use the real position reports recorded by the Australian Advanced Air Traffic System (TAAATS) used for current air traffic management in Australia (McFadyen and Martin, 2016). As a pre-processing step the data is transformed into trajectories utilising the aircraft identification tags and reported timing. Trajectories with less than 3 position reports are then removed as they are too short for the trajectory modelling task. As the final step each flight trajectory is re-sampled to provide 50 equally spaced data points. This gives us 260,735 trajectories. The aircraft trajectories are represented as 3 dimensional data streams where each point  $\mathbf{x}_i^t$  of the input sequence  $X_i = [\mathbf{x}_i^1, \mathbf{x}_i^2, \dots, \mathbf{x}_i^T]$  can be represented as,

$$\mathbf{x}_i^t = \begin{pmatrix} x_i^t \\ y_i^t \\ z_i^t \end{pmatrix}, \quad (17)$$

where  $T$  is the number of time steps in the  $i^{th}$  input sequence. In this experiment we observed the first 25 frames of the aircraft trajectory and predicted the trajectory for next 25 frames. For training we selected the first 182,515 (i.e 70%) trajectories chronologically. The remaining 78,220 trajectories were used for testing.

Based on the recommendations provided in (Gong and McNally (2004); Ayhan and Samet (2016); Paglione and Oaks (2007)), we measure the following three error metrics for the aircraft trajectory prediction experiment. The trajectory prediction errors are calculated for each observed radar track point in each input trajectory segment. Let  $n$  be the number of trajectories in the testing set and we are interested in predicting the trajectory for the time period from  $t = T^{obs} + 1$  to  $T^{pred}$  while observing the trajectory of the same aircraft from  $t = 1$  to  $T^{obs}$ . Let the predicted course from North for the trajectory  $i$  at  $t^{th}$  time instance be denoted by  $\hat{\theta}_i^t$ . Then,

$$\Delta x_i^t = \hat{x}_i^t - x_i^t, \quad (18)$$

and,

$$\Delta y_i^t = \hat{y}_i^t - y_i^t. \quad (19)$$

Now we can define,

1. *Average along track error (AE):*

$$AE = \frac{\sum_{i=1}^n \sum_{t=T^{obs}+1}^{T^{pred}} (\Delta x_i^t \sin(\hat{\theta}_i^t) + \Delta y_i^t \cos(\hat{\theta}_i^t))}{n(T^{pred} - (T^{obs} + 1))} \quad (20)$$

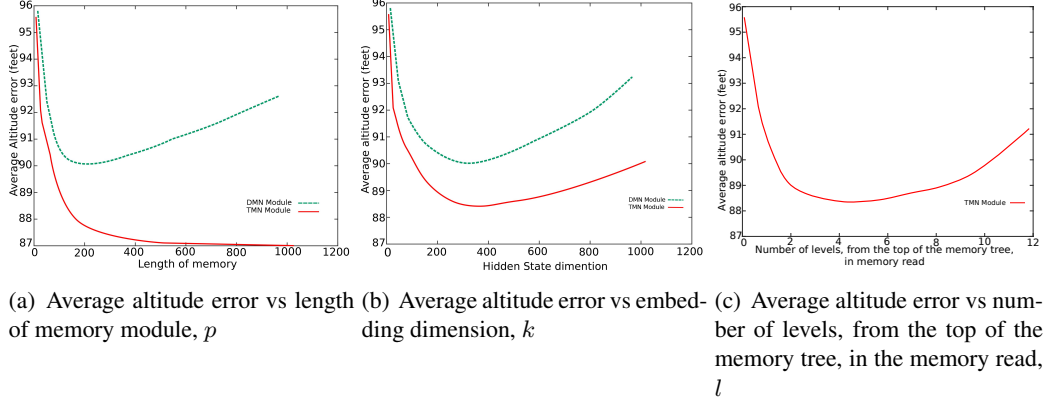


Figure 5: Parameter evaluation for length of memory module,  $p$ , embedding dimension,  $k$ , and the number of levels, from the top of the memory tree, in the memory read,  $l$

2. Average cross track error (CE):

$$CE = \frac{\sum_{i=1}^n \sum_{t=T^{obs}+1}^{T^{pred}} (\Delta x_i^t \cos(\hat{\theta}_i^t) - \Delta y_i^t \sin(\hat{\theta}_i^t))}{n(T^{pred} - (T^{obs} + 1))} \quad (21)$$

3. Average altitude error (ALE):

$$ALE = \frac{\sum_{i=1}^n \sqrt{\sum_{t=T^{obs}+1}^{T^{pred}} (\hat{z}_i^t - z_i^t)^2}}{n(T^{pred} - (T^{obs} + 1))} \quad (22)$$

As baseline models we implemented the HMM model (**HMM**) proposed in Ayhan and Samet (2016) and the Dynamic Memory Networks approach (**DMN**) given in Kumar et al. (2015). For the **HMM** model the required weather data is obtained from the data services provided by the Australian Bureau of Meteorology (BOM (November 2016)). We observed wind speed and direction and temperature at one minute frequency. The data interpolation and parameter quantisation is performed in the same way as in Ayhan and Samet (2016).

#### 4.1.1 PREDICTION UNDER NORMAL CONDITIONS

Hyper parameters, the length of the memory module,  $p$ , and the embedding dimension,  $k$ , of the proposed memory module (**TMN**) and **DMN** are evaluated experimentally. Fig. 5 (a) shows the variation in average altitude error against  $p$  for the **TMN** and for **DMN** modules in solid red and dashed green lines respectively. For the proposed **TMN**, as the error converges around  $p = 500$ , we set the value of  $p$  as 512. For **DMN** the plot shows that error decreases at first and it starts increasing again when the length of the memory exceeds 200 hidden units. This verifies our assertion that naive memory models with sequential LSTM architectures fail to model long term dependencies.

As  $p = 180$  gives the lowest altitude error, for the **DMN** model we set the value of  $p$  as 180. With a similar experiment we evaluated the optimal embedding dimension,  $k$ . Fig. 5 (b) shows the variation of average altitude error against  $k$  for the **TMN** and for **DMN** modules. As for both modules  $k = 300$  produces the smallest altitude error, we set the embedding dimension to 300 units.

Finally, we evaluated the number of levels from the top of the memory tree in the memory read,  $l$ , of the proposed memory module (**TMN**). The evaluation results, shown in Fig. 5 (c), suggest that  $l = 4$  produces optimal results. It is observed that the error is reduced until  $l = 4$  before increasing again when the number of levels in the memory read operation exceeds 6 levels. This is due to the density of the extracted memory activation. Using the tree structure of the memory we capture the information in a hierarchical manner, where only vital information from the bottom layers is passed to the top layer. Therefore when the extracted matrix becomes too dense, the decoding function fails to discriminate such vital facts, and the performance starts to degrade.

We trained the TMN model using stochastic gradient descent (SGD) with momentum and evaluation results are presented in Table 1.

Metirc	HMM	DMN	TMN
AE	1.103	1.039	<b>1.020</b>
CE	1.042	1.056	<b>1.011</b>
ALE	147.801	92.039	<b>87.001</b>

Table 1: Quantitative results for aircraft trajectory prediction. In all the methods the forecast trajectories are of length 25 frames. The first row reports the along track error (AE), the second row shows cross track error (CE) and the final row shows the altitude error (ALE).

With reference to the results in Table 1 we illustrate the capability of the proposed model to infer different modes of air traffic behaviour. We would like to emphasise that without explicit weather modelling or neighbourhood modelling the proposed augmented memory architecture is capable of learning the salient aspects and long term dependencies which are necessary when modelling the aircraft trajectories. Another prominent factor is the accuracy improvement from the **DMN** model to the **TMN** model which illustrates that flat memory architectures such as that of Kumar et al. (2015) fail to capture long term dependencies; where as our proposed hierarchical memory architecture is able to successfully learn those long term relationships. In Section 5 we perform an in depth analysis on the hidden state activations of the memory module which illustrates how the proposed multi-layer architecture generates future trajectories while encoding the necessary information from the history.

In Fig. 6 we show prediction results of the **HMM** model, **DMN** model and our model (**TMN**) on the aircraft trajectory dataset. It should be noted that our model generates better predictions even with the highly varied nature of the database (i.e take off, landing, cruising, etc). For instance in the 1st and 2nd rows we demonstrate how the same model adapts to takeoff and landing scenarios. In the last row of Fig. 6 we show some failure cases. The reason for such deviations from the ground truth were mostly due to sudden turns and movements. Even though these trajectories do not match the ground truth, the proposed method still outperforms the current state-of-the-art methods and generates much more realistic trajectories.

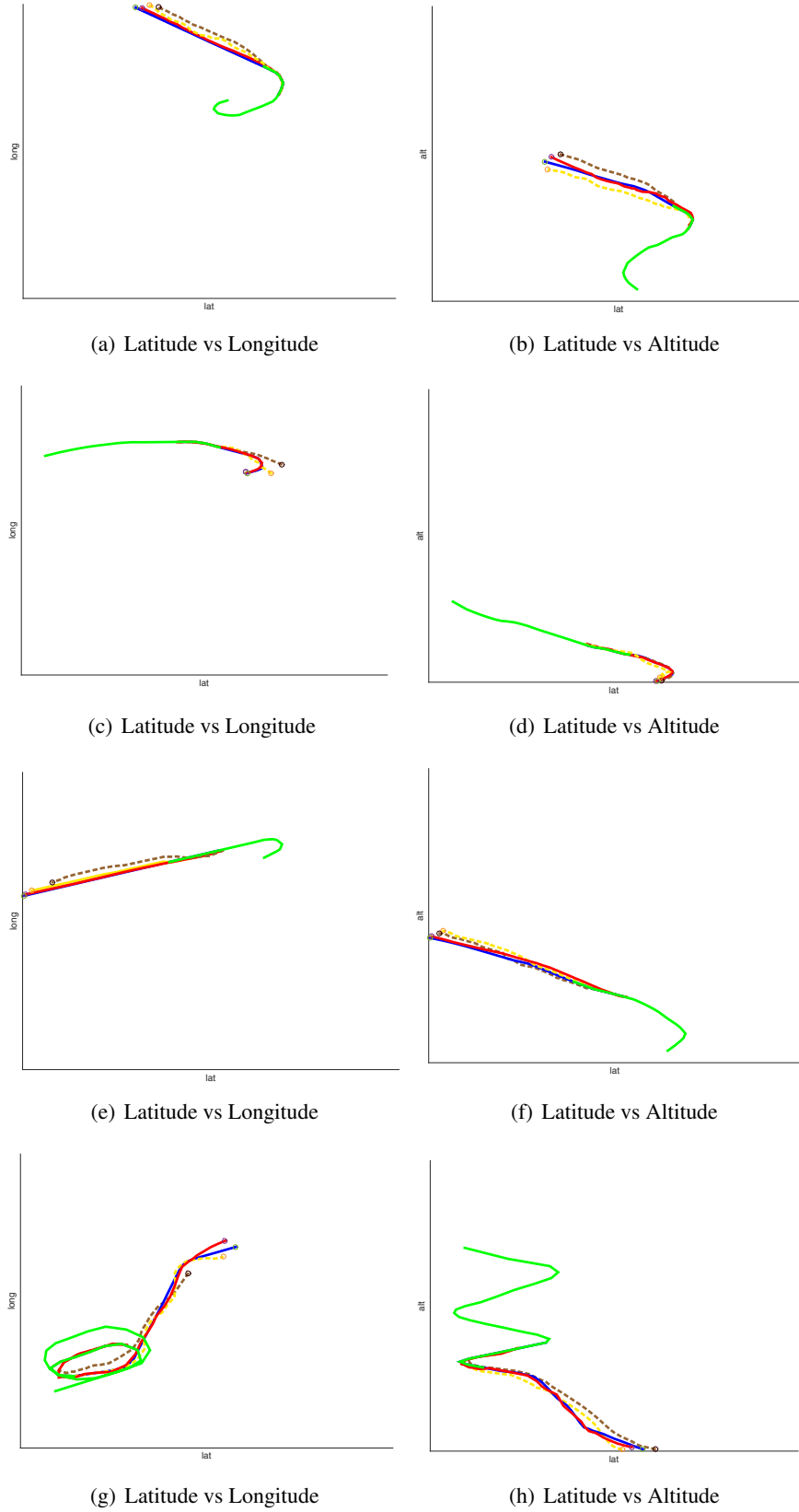


Figure 6: Qualitative results under nominal conditions: Each row represents a particular example. Given (in green), Ground Truth (in Blue) and Predicted trajectories from the **TMN** model (in red), from the **DMN** model (in yellow), from the **HMM** model (in brown).

#### 4.1.2 HANDLING DIFFERENT WEATHER CONDITIONS

In order to verify the capability of the proposed model to understand and capture the effect of weather on aircraft trajectory prediction via historical trajectories we conducted a separate experiment, where the model is used to predict the air traffic on a stormy day. Severe storms affected South East Queensland on 28th of November 2015. We tested the model with the trajectories from 27th November to 29th November (i.e. 1916 trajectories) as it allows a sufficient number of examples to initialise the memory module. It should be noted that all these examples are not used for training the model. We compare the proposed model against the **HMM** model (Ayhan and Samet (2016)) which is explicitly fed with the weather information.

Metirc	HMM	TMN
AE	1.689	<b>1.146</b>
CE	1.967	<b>1.513</b>
ALE	203.754	<b>88.397</b>

Table 2: Quantitative results for aircraft trajectory prediction under stormy conditions. In all the methods the forecast trajectories are of length 25 frames. The first row reports the along track error (AE), the second row shows cross track error (CE) and the final row shows the altitude error (ALE).

When comparing Table. 2 with Table. 1, though the accuracy of the predictions of the **TMN** is slightly reduced, the predictions are still more accurate when compared with Ayhan and Samet (2016) in which the error has increased dramatically indicating that the baseline model does not adapt well to changing weather conditions.

When referring to the results presented in Fig. 7 it is evident that the non uniform nature of the air traffic under stormy weather conditions is well captured by the proposed approach. Even with the accurate weather information, the **HMM** model fails to properly exploit this data and generates erroneous trajectories. The **TMN** model anticipates the uneven nature through looking at the recent history while also mapping how correlated trajectories have behaved when considering the long term history.

Furthermore, we would like to emphasise the fact that for the **TMN** model, even under storm conditions, the altitude error is within the  $\pm 100$ ft altimeter tolerances provided to private pilots (AIP (2017)).

## 4.2 Experiment 2: Pedestrian trajectory prediction

In this experiment we considered 3 months worth of trajectories from the Edinburgh Informatics Forum database (Majecka (2009)), where we trained our model on 60,000 trajectories and tested on 12,000 trajectories. For all the models we observed the trajectory for 30 frames and predicted the trajectory for the next 30 frames. In this experiment the trajectories are represented as 2 dimensional data where each point  $\mathbf{x}_i^t$  of the input sequence  $X_i = [\mathbf{x}_i^1, \mathbf{x}_i^2, \dots, \mathbf{x}_i^T]$  is represented as,

$$\mathbf{x}_i^t = \begin{pmatrix} x_i^t \\ y_i^t \end{pmatrix}. \quad (23)$$

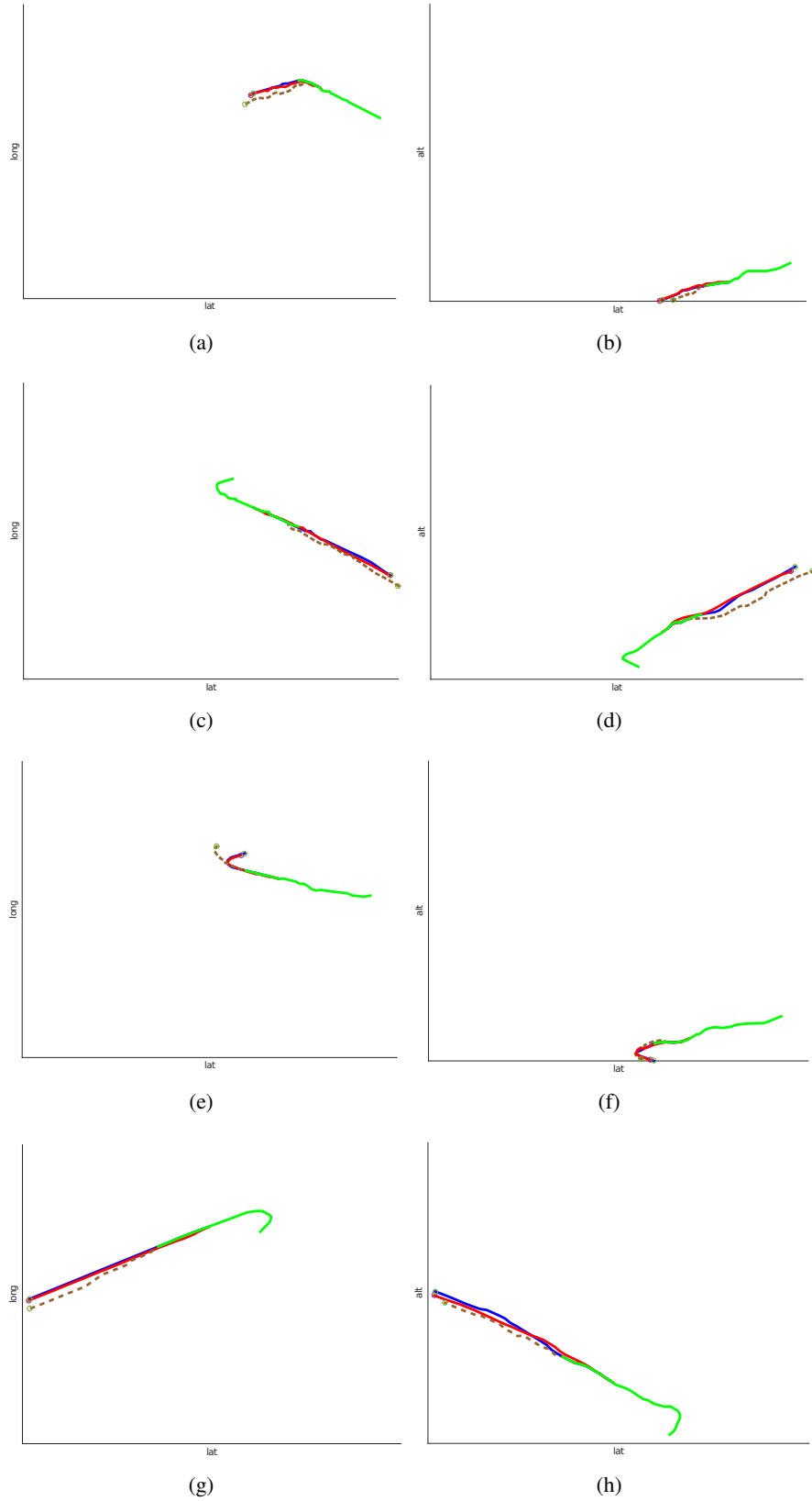


Figure 7: Qualitative results under storm conditions: Each row represents a particular example. Given (in green), Ground Truth (in Blue) and Predicted trajectories from **TMN** model (in red), from **HMM** model (in brown).



For comparison we implemented Soft+hard wired attention (**SH-Atn**) model from Fernando et al. (2017b), the Social LSTM (**So-LSTM**) model given in Alahi et al. (2016) and Dynamic Memory Networks (**DMN**) given in Kumar et al. (2015). For the **So-LSTM** model a local neighbourhood of size 32px was considered and the hyper-parameters were set according to Alahi et al. (2016). For Fernando et al. (2017b) we considered a neighbourhood size of 10 in the left, right and front directions and an embedding size of 300 hidden units. For **DMN** and **TMN** models we use the same experimental settings given in the previous experiment. Similar to (Fernando et al. (2017b); Alahi et al. (2016)) we report prediction accuracy with the following 3 error metrics. We are predicting a trajectory for the period from  $t = T^{obs} + 1$  to  $T^{pred}$  while observing the same trajectory from  $t = 1$  to  $T^{obs} + 1$ . Let  $n$  be the number of trajectories in the testing set and  $\hat{X}_i^t$  be the predicted position for the trajectory  $i$  at the  $t^{th}$  time instance, and  $X_i^t$  be the respective observed positions then,

1. *Average displacement error (ADE):*

$$ADE = \frac{\sum_{i=1}^n \sum_{t=T^{obs}+1}^{T^{pred}} (\hat{X}_i^t - X_i^t)^2}{n(T^{pred} - (T^{obs} + 1))}. \quad (24)$$

2. *Final displacement error (FDE) :*

$$FDE = \frac{\sum_{i=1}^n \sqrt{(\hat{X}_i^{T^{pred}} - X_i^{T^{pred}})^2}}{n}. \quad (25)$$

3. *Average non-linear displacement error (n-ADE):* The average displacement error for the non-linear regions of the trajectory.

$$n - ADE = \frac{\sum_{i=1}^n \sum_{t=T^{obs}+1}^{T^{pred}} I(\hat{X}_i^t)(\hat{X}_i^t - X_i^t)^2}{\sum_{i=1}^n \sum_{t=T^{obs}+1}^{T^{pred}} I(\hat{X}_i^t)}, \quad (26)$$

where

$$I(\hat{X}_i^t) = \begin{cases} 1 & \text{if } \frac{d^2 y_i^t}{d(x_i^t)^2} \neq 0. \\ 0 & \text{o.w} \end{cases} \quad (27)$$

As shown in Table 3, the proposed model outperforms the **SH-Atn** model, **So-LSTM** model and **DMN** in all three error metrics. The dataset is considered quite challenging where there are multiple source and sink positions, different crowd motion patterns are present and motion paths are heavily crowded. Without explicitly modelling the neighbourhood motion such as in the SH-Atn model and So-LSTM model, our approach is still outperforming the current state-of-the-art models.

In Fig. 8 we show prediction results for the **SH-Atn** model, **DMN** model and our model **TMN** on the EIF trajectory dataset. With the examples shown it is evident how different modes of human motion are well captured and represented through the proposed augmented memory module. For instance in Fig. 8 (g) and Fig. 8 (k) the pedestrians exhibit a sudden change in motion which all baseline models fail to capture. But the proposed model has successfully anticipated that motion via historical behaviour.

Metirc	SH-Atn	So-LSTM	DMN	TMN
ADE	1.066	1.843	1.798	<b>1.051</b>
FDE	1.551	2.421	2.276	<b>1.398</b>
n-ADE	1.021	1.988	1.456	<b>0.987</b>

Table 3: Quantitative results for pedestrian trajectory prediction. In all the methods the forecast trajectories are of length 30 frames. The first row represents the average displacement error (ADE), the second row shows the final displacement error (FDE) and the third row shows the average non-linear displacement error (n-ADE).

## 5. Analysis of Memory Activations

In this section we explore how elements of the memory module are activated under different conditions. In particular, we aim to show that the baseline model Kumar et al. (2015) and the first layer of the proposed approach have activations based more on the recent inputs to the model; while the last layer of the proposed approach has activations that are driven by the input itself rather than the recent history, due to it’s ability to better capture the long term dependencies. We conducted this investigation with the pedestrian dataset as it is simpler to visualise.

### 5.1 Correlated activations from the first layer

In this section we analyse the memory activation for various test cases from Experiment 2. We randomly selected a memory cell from the first layer of the memory, and analysed the temporal evaluation of the activations of the hidden states of that particular memory cell. Each memory cell has a hidden state dimension of 100 and every input sequence has a length of 30 time steps. A smaller hidden state dimension is used compared to previous experiments to ensure the clear visibility of memory activations. Fig. 9 shows the results of our analysis process. We searched our test set for common activation patterns. The first column of Fig. 9 shows the memory activations, with the most highly correlated memory patterns shown in red, green and blue; and the remainder of the activations shown in grey. The second column shows the input to the model (in green) and the predicted sequence (in blue). The previous 10 trajectories that are fed to the memory are shown in the third column. From black to white we have colour coded the most recent trajectory to the oldest trajectory. We expect the first layer of the memory module to generate similar activation patterns when there exists a similar set of historical trajectories, and if there exists a similarity between those historical trajectories and the input.

### 5.2 Correlated activations from the last layer

Fig. 10 illustrates the activations of final memory cell (i.e last layer). Column descriptions are identical to the that of Fig. 9. The second and third columns of Fig. 10 provide visual evidence that the proposed memory module has successfully learnt relationships among input trajectories. If our memory module has enough capacity and if it is capturing long term dependencies, the final layer should generate similar activations for similar input trajectory patterns, rather than being completely reliant on the current short term context. That is evident with the similarity shown in Fig. 10 where we observe similar activations (i.e column 1) for similar inputs patterns (i.e column 2). Importantly,

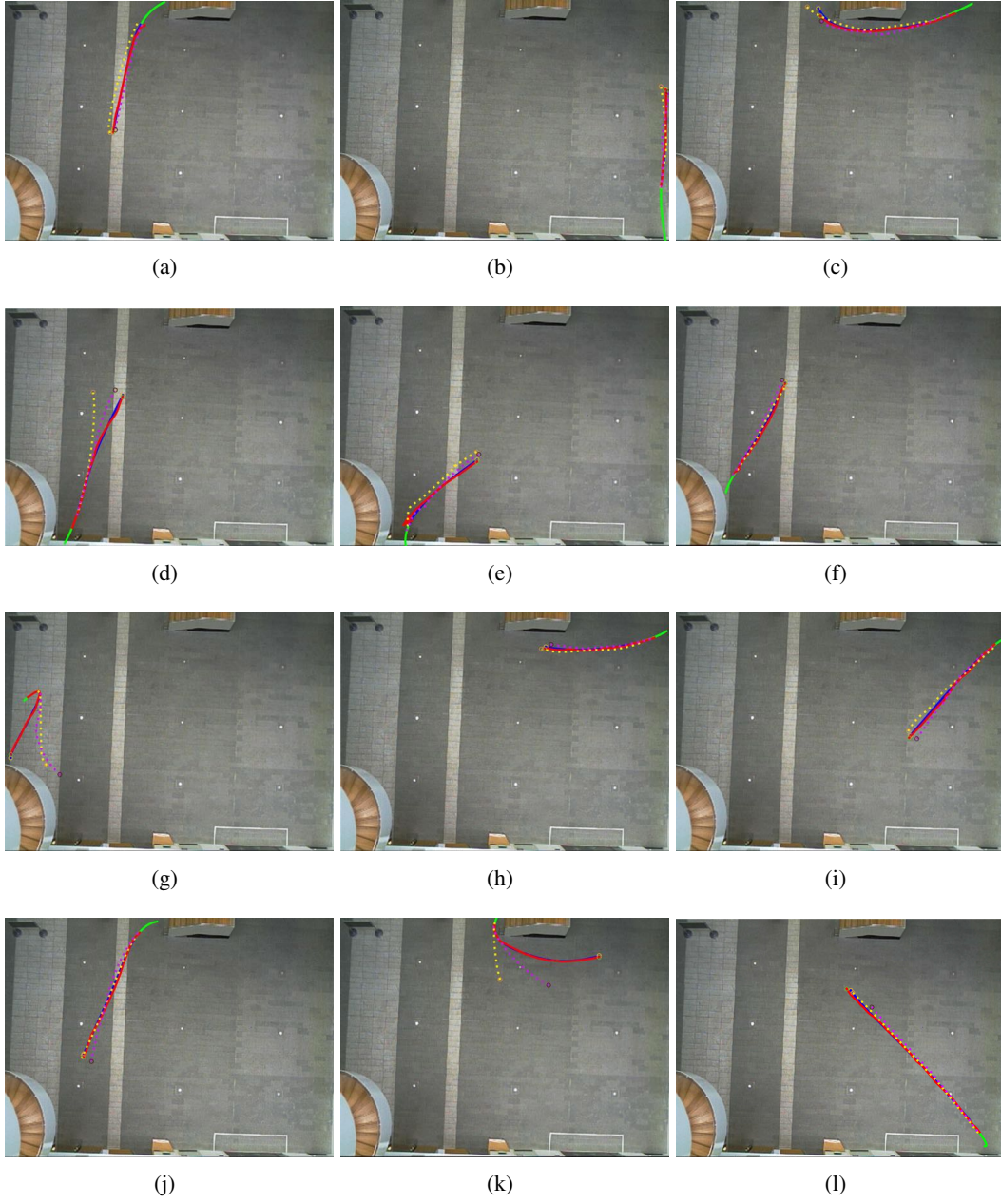


Figure 8: Qualitative results: Given (in green), Ground Truth (in Blue) and Predicted trajectories from **TMN** model (in red), from **DMN** model (in yellow), from **SH-Atn** model (in purple).

we note that despite the similar activations, the recent history (i.e. column 3) differs significantly between cases; unlike in Fig. 9.

### 5.3 Activations from the baseline memory module (Kumar et al. (2015))

In Fig. 11 we visualise the correlated activations from the baseline memory model shown in Fig. 3 (a). The column labels are identical to that of Fig. 9. We compare the activations in Fig. 10 to those in Fig. 11. When observing Fig 11 column 3 it is evident that hidden state activations are dominated by the most recent inputs to the memory, and the long term dependencies are of little importance. This is noted to be an inherent problem with sequential LSTM architectures (Chen et al. (2016)). Therefore regardless of the input to the model (shown in Fig 11 column 2) the memory module is generating similar activations only considering the short term context. Hence the prediction error is high. Furthermore we would like to highlight the similarity between Fig 11 and the first layer of the proposed memory module (shown in Fig. 9 ).

To further illustrate the limitations of the baseline model, we clustered the input trajectories and from one particular cluster we extracted out the highest correlated input trajectories (shown in Fig. 12) within that cluster. Even though we expect the baseline memory module to generate similar activations when the inputs to the prediction module are similar, it generates vastly different activations. In order to highlight the differences among memory activations we randomly selected 3 hidden units within the memory and illustrated their temporal evaluations in colour (see Fig. 12 column 1). The rest of the activations are shown in grey. As the short term history within the baseline memory module is significantly different from one example to another (see Fig. 12 column 3) the memory module generates extremely different activations, completely discarding the input given. Furthermore when comparing Fig. 12 with Fig. 11 the baseline memory module generates similar activations (shown in Fig. 11 column 1) when the short term history is similar (shown in Fig. 11 column 3) and vastly different activations (shown in Fig. 12 column 1) when short term history is different (shown in Fig. 12 column 3). The input to the prediction module and its long term dependencies are of little importance.

## 6. Conclusion

The proposed tree memory network (TMN) model is a generalised architecture for modelling long term and short term relationships, which can be applied directly for any sequence-to-sequence mapping task. With the evaluation results we demonstrated that our proposed memory architecture is able to outperform all the considered baselines and we provide visual evidence on the power of TMN, which is able to capture both long term and short term relationships via an efficient tree structure. We have demonstrated our approach on two different application scenarios from the trajectory prediction domain; however, the varied nature of these problems demonstrates how the proposed TMN model can be directly applied to any sequence-to-sequence prediction problem where modelling long term relationships is necessary. In future work we will be exploring the applications of TMN as an encoding mechanism for large scale multimodal inputs such as videos (i.e. a sequence of images), where the encoded vector representation of the memory can be utilised to generate a sparse representation of the entire video sequence with its temporal relationships.

## References

AIP. Altimeter checks and flight tolerances, aairservices, australia. *Aeronautical Information Package*, page 39, 2017.

- Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. *CVPR*, 2016.
- Samet Ayhan and Hanan Samet. Aircraft trajectory prediction made easy with predictive analytics. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 21–30, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939694.
- Rafael E Banchs. A principled approach to context-aware machine translation. *EACL 2014*, pages 70–74, 2014.
- BOM. Australian bureau of meteorology,. November 2016. URL <http://www.bom.gov.au/climate/data/stations/>.
- Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2722–2730, 2015.
- Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. Enhancing and combining sequential and tree lstm for natural language inference. *arXiv preprint arXiv:1609.06038*, 2016.
- Xinlei Chen and C Lawrence Zitnick. Learning a recurrent visual representation for image caption generation. *arXiv preprint arXiv:1411.5654*, 2014.
- Patrick Pakyan Choi and Martial Hebert. Learning and predicting moving object trajectory: a piecewise trajectory segment approach. *Robotics Institute*, page 337, 2006.
- Rod Cole, Steve Green, Matt Jardin, Barry Schwartz, and Stan Benjamin. Wind prediction accuracy for air traffic management decision support tools. In *3rd USA/Europe Air Traffic Management R&D Seminar, Napoli, Italy*, 2000.
- AMP de Leege, MM Van Paassen, and M Mulder. A machine learning approach to trajectory prediction. 2013.
- D Delahaye. *Wind field update using radar track data*. PhD thesis, Masters thesis, Ecole Nationale de l’Aviation Civile, 1992.
- Rmi Emonet, Jagannadan Varadarajan, and Jean-Marc Odobez. Extracting and locating temporal motifs in video scenes using a hierarchical non parametric bayesian model. In *CVPR*, pages 3233–3240, 2011. ISBN 978-1-4577-0394-2.
- Tharindu Fernando, Simon Denman, Sridha Sridharan, and Clinton Fookes. Deep decision trees for discriminative dictionary learning with adversarial multi-agent trajectories. *under review as a journal paper in IEEE Transactions on Knowledge and Data Engineering*, 2017a.
- Tharindu Fernando, Simon Denman, Sridha Sridharan, and Clinton Fookes. Soft+ hardwired attention: An lstm framework for human trajectory prediction and abnormal event detection. *arXiv preprint arXiv:1702.05552*, 2017b.
- Chester Gong and Dave McNally. A methodology for automated trajectory prediction analysis. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, pages 16–19, 2004.

- Dirk Helbing and Péter Molnár. Social force model for pedestrian dynamics. *Phys. Rev. E*, 51: 4282–4286, 1995. doi: 10.1103/PhysRevE.51.4282.
- WM Hollister, ER Bradford, and JD Welch. Using aircraft radar tracks to estimate wind aloft. *The Lincoln Laboratory Journal*, 2(3):555–565, 1989.
- Timothy M. Hospedales, Shaogang Gong, and Tao Xiang. A markov clustering topic model for mining behaviour in video. In *ICCV*, pages 1165–1172, 2009. ISBN 978-1-4244-4419-9.
- Armand Joulin and Tomas Mikolov. Inferring algorithmic patterns with stack-augmented recurrent nets. In *Advances in Neural Information Processing Systems*, pages 190–198, 2015.
- Kshitij Judah, Alan P Fern, Thomas G Dietterich, et al. Active Imitation learning: formal and practical reductions to iid learning. *The Journal of Machine Learning Research*, 15(1):3925–3963, 2014.
- Łukasz Kaiser and Ilya Sutskever. Neural gpu learn algorithms. *arXiv preprint arXiv:1511.08228*, 2015.
- Hema Swetha Koppula and Ashutosh Saxena. Anticipating human activities using object affordances for reactive robotic response. In *IROS*, page 2071, 2013. ISBN 978-981-07-3937-9.
- Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285*, 2015.
- John Lafferty, Andrew McCallum, Fernando Pereira, et al. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, volume 1, pages 282–289, 2001.
- Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*, 2015.
- HaiLong Liu, Tadahiro Taniguchi, Yusuke Tanaka, Kazuhito Takenaka, and Takashi Bando. Visualization of driving behavior based on hidden feature extraction by using deep learning. *IEEE Transactions on Intelligent Transportation Systems*, 2017.
- B. Majecka. Statistical models of pedestrian behaviour in the forum. *MSc Dissertation, School of Informatics, University of Edinburgh*, 2009.
- Mateusz Malinowski and Mario Fritz. A multi-world approach to question answering about real-world scenes based on uncertain input. In *Advances in Neural Information Processing Systems*, pages 1682–1690, 2014.
- Aaron McFadyen and Terry Martin. Terminal airspace modelling for unmanned aircraft systems integration. In *Unmanned Aircraft Systems (ICUAS), 2016 International Conference on*, pages 789–794. IEEE, 2016.
- Tomas Mikolov, Armand Joulin, Sumit Chopra, Michael Mathieu, and Marc’Aurelio Ranzato. Learning longer memory in recurrent neural networks. *arXiv preprint arXiv:1412.7753*, 2014.

- Stephane Mondoloni and Dianna Liang. Improving trajectory forecasting through adaptive filtering technique. In *Proceedings of 5th USA-Europe ATM Seminar*, 2003.
- Tsendsuren Munkhdalai and Hong Yu. Neural semantic encoders. *arXiv preprint arXiv:1607.04315*, 2016a.
- Tsendsuren Munkhdalai and Hong Yu. Neural tree indexers for text understanding. *arXiv preprint arXiv:1607.04492*, 2016b.
- Arvind Neelakantan, Quoc V Le, and Ilya Sutskever. Neural programmer: Inducing latent programs with gradient descent. *arXiv preprint arXiv:1511.04834*, 2015.
- Mike M Paglione and Robert D Oaks. Implementation and metrics for a trajectory prediction validation methodology. In *Proceedings of the American Institute of Aeronautics and Astronautics (AIAA) Guidance, Navigation, and Control Conference, Hilton Head, South Carolina*, 2007.
- Russell A Paielli and Heinz Erzberger. Conflict probability estimation for free flight. *Journal of Guidance, Control, and Dynamics*, 20(3):588–596, 1997.
- Stefano Pellegrini, Andreas Ess, and Luc J. Van Gool. Improving data association by joint modeling of pedestrian trajectories and groupings. In *ECCV*, pages 452–465, 2010. ISBN 978-3-642-15548-2.
- Maria Prandini, Jianghai Hu, John Lygeros, and Shankar Sastry. A probabilistic approach to aircraft conflict detection. *IEEE Transactions on intelligent transportation systems*, 1(4):199–220, 2000.
- Scott Reed, Zeynep Akata, Xinchun Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 3, 2016.
- CM Rekkas, CC Lefas, and NJ Krikelis. Three-dimensional tracking using on-board measurements. *IEEE transactions on aerospace and electronic systems*, 27(4):617–624, 1991.
- Iulian Vlad Serban, Alessandro Sordani, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. A hierarchical latent variable encoder-decoder model for generating dialogues. *arXiv preprint arXiv:1605.06069*, 2016.
- Elaheh ShafieiBavani, Mohammad Ebrahimi, Raymond Wong, and Fang Chen. An efficient approach for multi-sentence compression. In *JMLR: Workshop and Conference Proceedings*, 2016.
- Xiaogang Wang, Keng Teck Ma, Gee Wah Ng, and W. Eric L. Grimson. Trajectory analysis and semantic region modeling using a nonparametric bayesian model. In *CVPR*, pages 1–8, 2008. ISBN 978-1-4244-2242-5.
- Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.
- Lee Francis Winder. *Hazard avoidance alerting with Markov decision processes*. PhD thesis, Massachusetts Institute of Technology, 2004.

- Jiacheng Xu, Danlu Chen, Xipeng Qiu, and Xuangjing Huang. Cached long short-term memory neural networks for document-level sentiment classification. *arXiv preprint arXiv:1610.04989*, 2016.
- Jingxin Xu, Simon Denman, Clinton B. Fookes, and Sridha Sridharan. Unusual scene detection using distributed behaviour model and sparse representation. *AVSS*, pages 48 – 53, 2012.
- Kota Yamaguchi, Alexander C. Berg, Luis E. Ortiz, and Tamara L. Berg. Who are you with and where are you going? In *CVPR*, pages 1345–1352, 2011. ISBN 978-1-4577-0394-2.
- Shuai Yi, Hongsheng Li, and Xiaogang Wang. Understanding pedestrian behaviors from stationary crowd groups. In *CVPR*, pages 3488 – 3496, 2015.
- Bolei Zhou, Xiaoou Tang, and Xiaogang Wang. Learning collective crowd behaviors with dynamic pedestrian-agents. *Journal of Computer Vision*, 111:50–68, 2015.
- Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. Long short-term memory over tree structures. arxiv preprint. *arXiv preprint arXiv:1503.04881*, 2015.



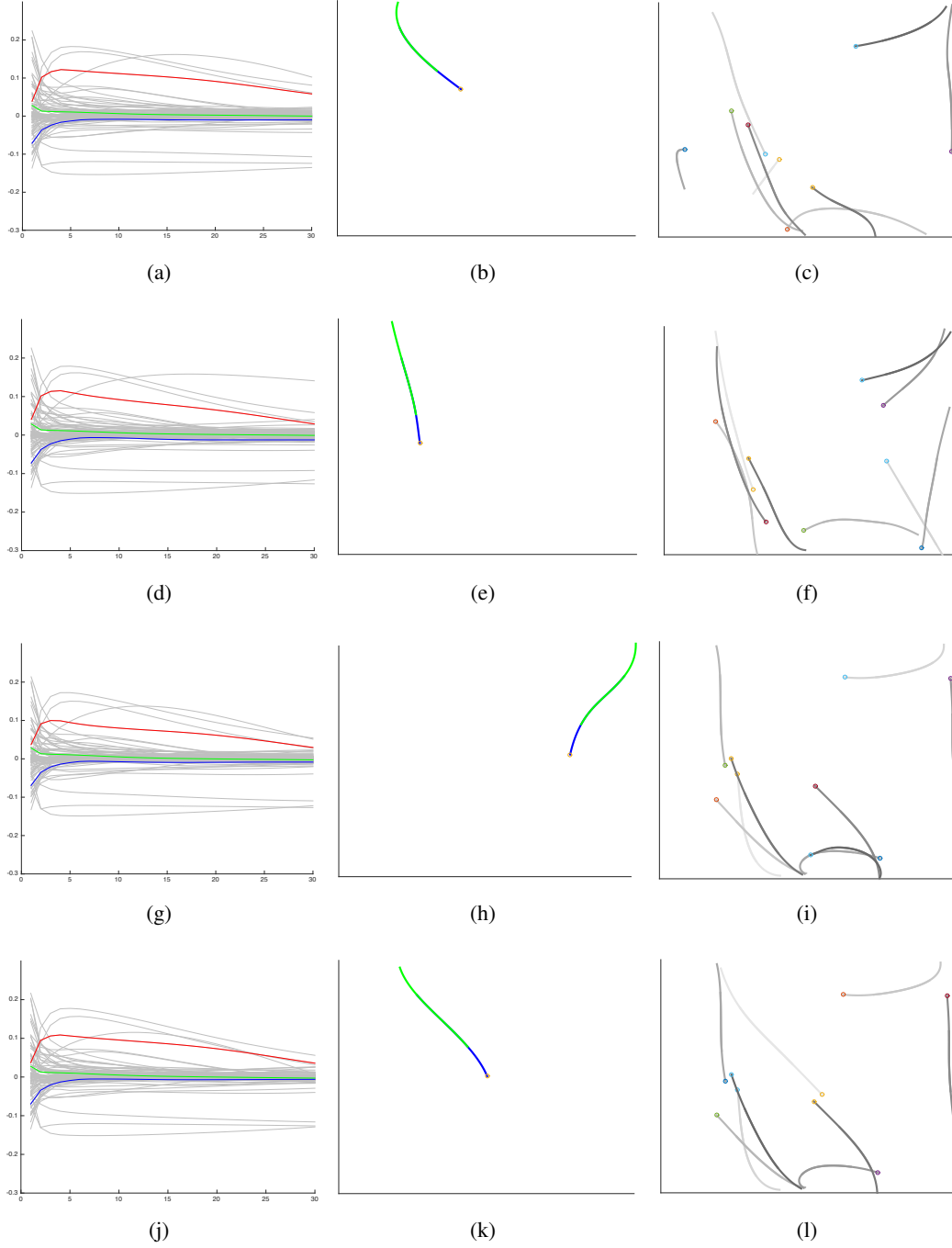


Figure 9: Pedestrian trajectories: Correlated activations from the first layer of the memory. First column: Highest correlated Memory activations for the pattern selected (in colours) and the rest of the activations (in grey) over time. Second column: The input (observed (in green) and predicted (in blue)) to the model at that time step. Third column: Previous 10 trajectories that reside in the memory. Black to white represents most recent to oldest.

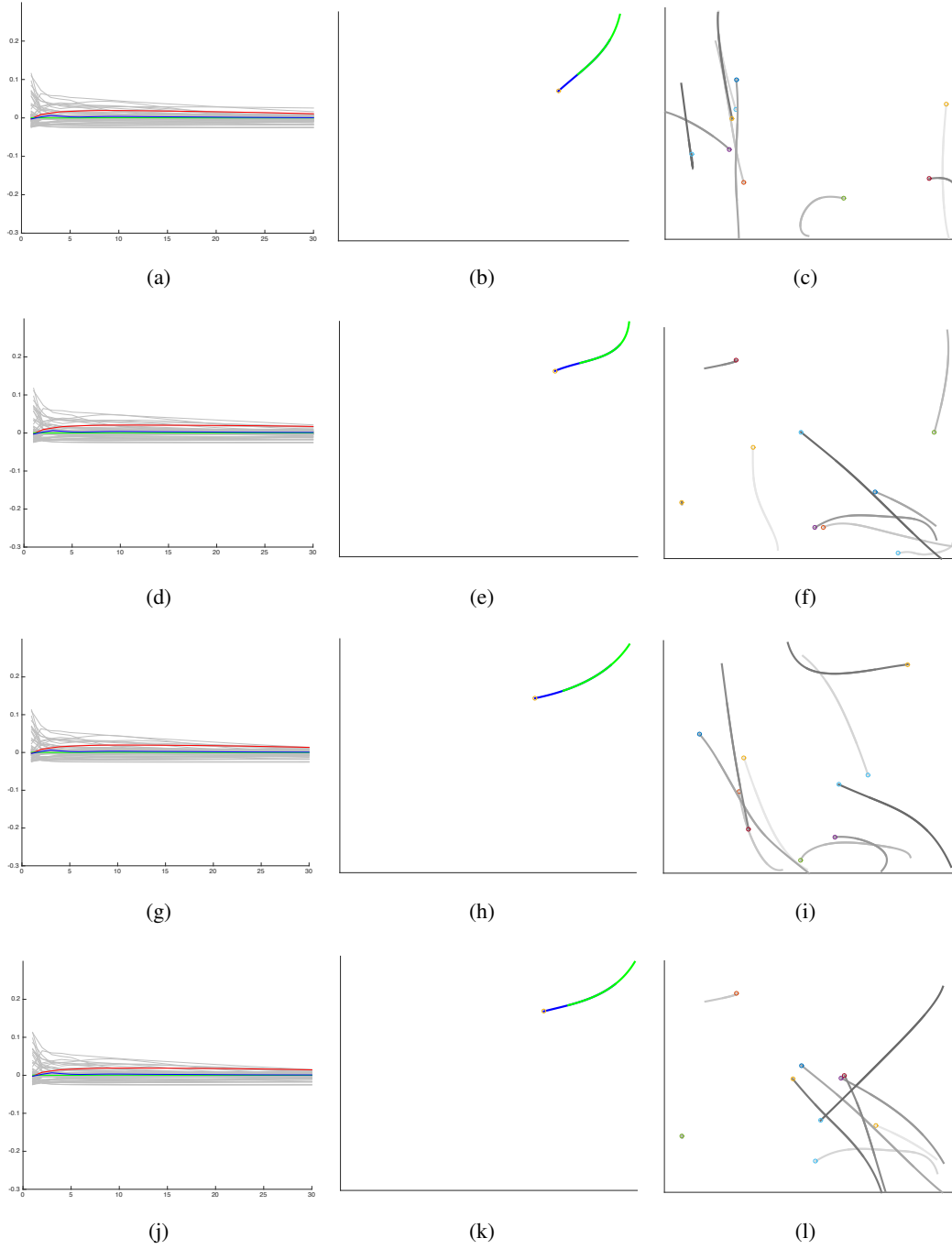


Figure 10: Pedestrian trajectories: Correlated activations from the last layer of the memory. First column: Highest correlated Memory activations for the pattern selected (in colours) and the rest of the activations (in grey) over time. Second column: The input (observed (in green) and predicted (in blue)) to the model at that time step. Third column: Previous 10 trajectories that reside in the memory. Black to white represents most recent to oldest.

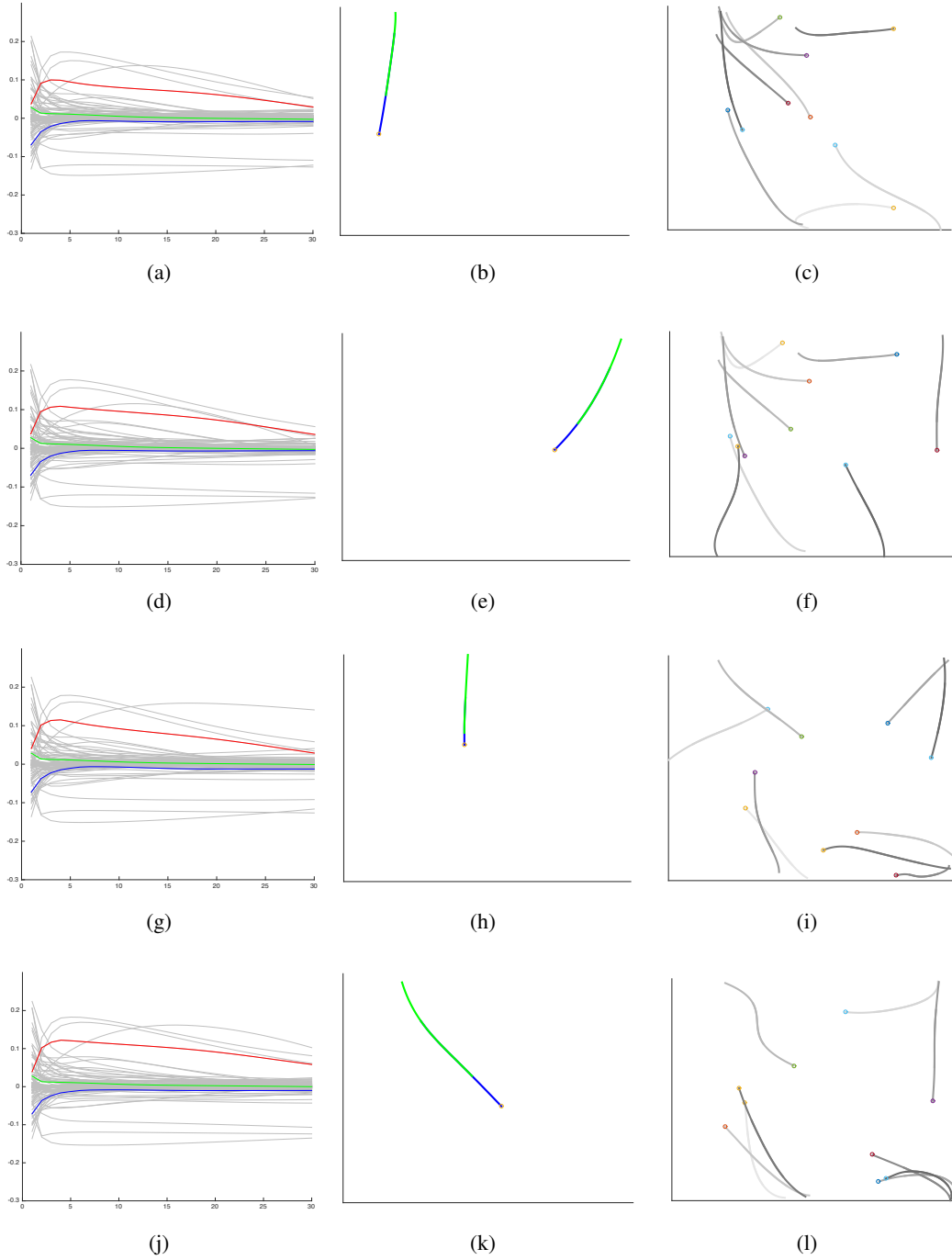


Figure 11: Pedestrian trajectories: Correlated activations from baseline memory model. First column: Highest correlated Memory activations for the pattern selected (in colours) and the rest of the activations (in grey) over time. Second column: The input (observed (in green) and predicted (in blue)) to the model at that time step. Third column: Previous 10 trajectories that reside in the memory. Black to white represents most recent to oldest.

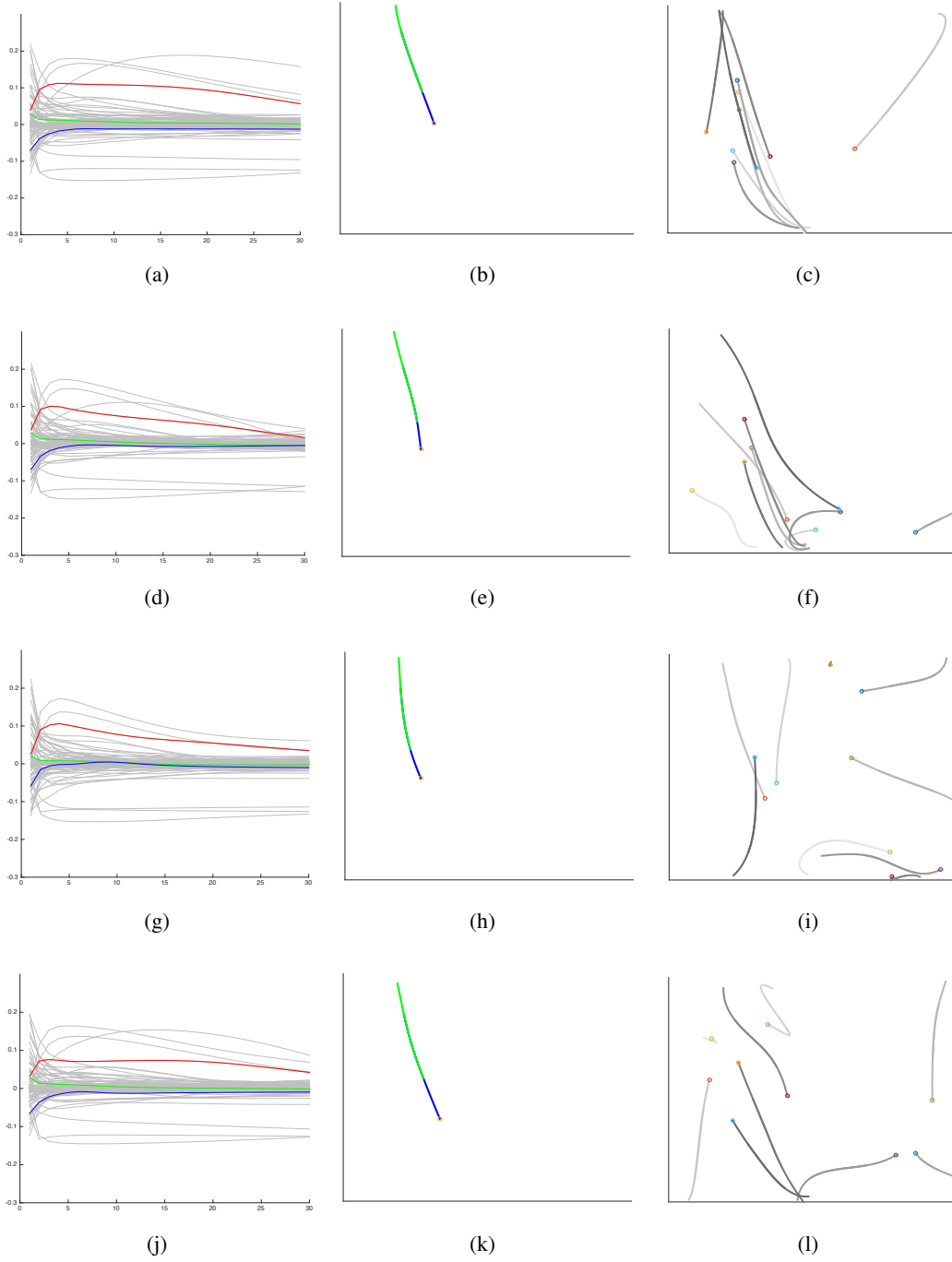


Figure 12: Pedestrian trajectories: Memory activations from baseline memory model for correlated inputs. First column: Memory activations over time, the pattern we are considering (in colours) and the rest of the activations (in grey). Second column: The correlated inputs (observed (in green) and predicted (in blue)). Third column: For the input selected in the second column, previous 10 trajectories that reside in the memory. Black to white represents most recent to oldest.